PDB_Assignment1_Exercise_2

2025-11-14

Name1: Adwitiya Argha Priyadarshini Boruah
Matriculation No. 1: 7070291
Email1: adbo00002@stud.uni-saarland.de

Name2: Md Mobashir Rahman
Matriculation No. 2: 7059086
Email2: mdra00001@stud.uni-saarland.de

Exercise 2.2: Detecting peaks in cell-cycle expression data

a.

The code begins by importing the expression dataset stored in a TSV file, where the first column contains gene identifiers and the remaining columns contain time-series expression measurements. The first column is renamed to ID to improve readability and ensure consistent referencing in later steps.

A subset of the dataset is then created by selecting the four assignment-specified genes (766, 932, 3297, 4685). For each selected gene, the corresponding row is extracted and converted into a numeric vector representing its expression profile across all time points.

To visualise the temporal expression patterns, the expression curve of gene 766 is plotted first, with axis limits determined from all four genes to maintain a common scale. The remaining gene profiles are then added to the same figure using additional line layers, each displayed in a different colour. Finally, a legend is included to indicate which colour corresponds to each gene, enabling straightforward interpretation of the combined time-series plot.

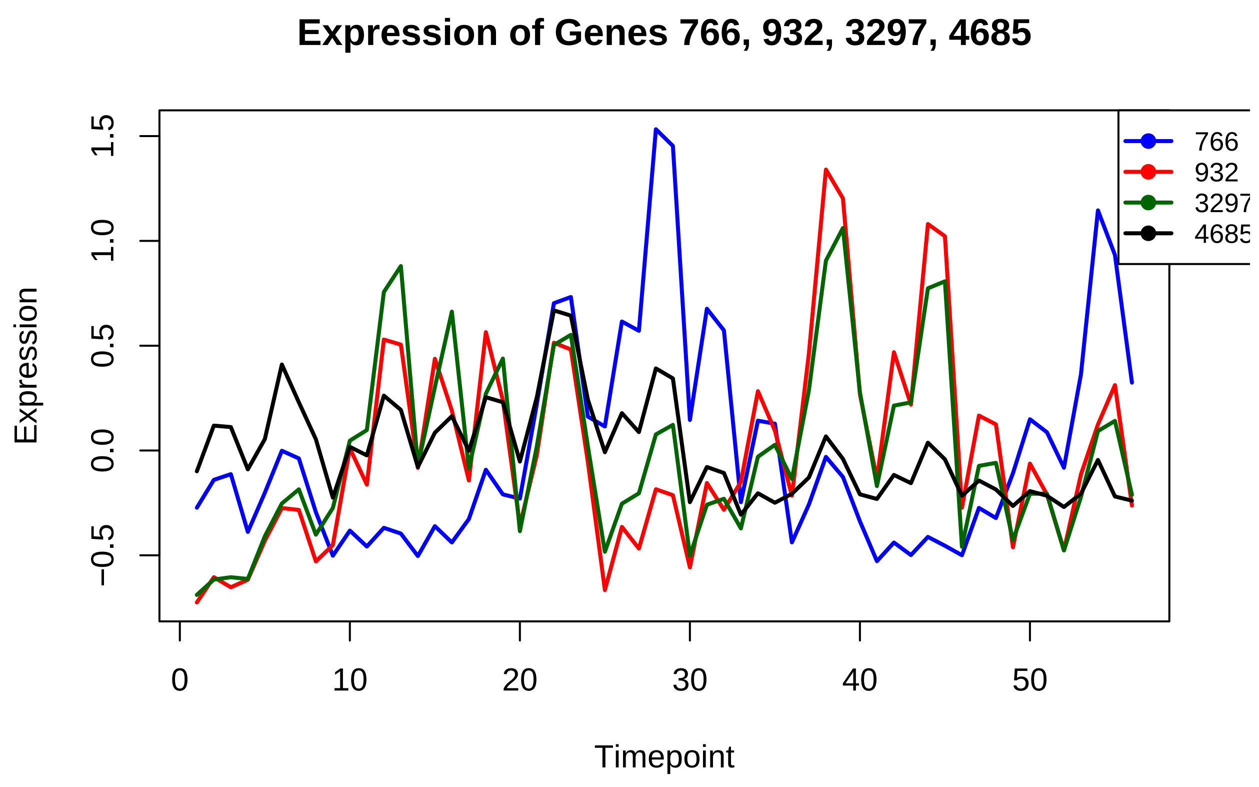
```r
# loading the expression data

#the dataset is stored as a TSV file where- the first column contains gene IDs, the remaining columns c

expression_data <- read.table("periodic_gene_expression.tsv",
                              header = TRUE,
                              sep = "\t",
                              check.names = FALSE)

# the first column corresponds to the gene identifiers.
#to make the later code readable, Renaming the first column to "ID"

colnames(expression_data)[1] <- "ID"


# selecting the 4 genes of interest

#these four specific genes are provided in the assignment.
```

```r
genes_of_interest <- c(766, 932, 3297, 4685)

#then extracting only the rows whose gene ID matches one of the four
selected_data <- expression_data[expression_data$ID %in% genes_of_interest, ]

# displaying the extracted rows to confirm successful selection
print(selected_data)
```

```
##        ID 0.132541748514212 0.22764074056482023 0.14828200919760418
## 766   766       -0.27288290          -0.1402047          -0.1128550
## 932   932       -0.72494565          -0.6044916          -0.6523827
## 3297 3297       -0.68898718          -0.6155137          -0.6046890
## 4685 4685       -0.09938045           0.1185993           0.1118007
##      0.0554196263670002 0.1734260578698282 0.21282449873561188
## 766         -0.38850498        -0.20155600        -0.001224659
## 932         -0.61646438        -0.43006450        -0.274523898
## 3297        -0.61280753        -0.40919111        -0.252765107
## 4685        -0.08997058         0.05437659         0.410260507
##      0.2959316374566157 0.10648439408557102 0.0532648666532511
## 766          -0.0377355         -0.29425663         -0.5019609
## 932          -0.2834915         -0.52907145         -0.4511528
## 3297         -0.1852222         -0.40165764         -0.2742209
## 4685          0.2298077          0.05389742         -0.2249058
##      0.09581446272390506 0.05129511406244348 0.07547365913936471
## 766          -0.38202540         -0.45801365          -0.3689231
## 932           0.01060914         -0.16274681           0.5293526
## 3297          0.04705137          0.09735093           0.7560012
## 4685          0.01736852         -0.02345934           0.2617990
##      0.1466597179944586         0.0 0.16170685189870834 0.0962233485286289
## 766          -0.3962889 -0.50365337         -0.36083764         -0.4387279
## 932           0.5050022 -0.08325892          0.43741532          0.1898679
## 3297          0.8799099 -0.07232278          0.30204116          0.6625337
## 4685          0.1937000 -0.07881011          0.08492213          0.1635400
##      0.221036209616606 0.2626756736252612 0.2617379530741627 0.0818723471906627
## 766       -0.326866497        -0.09191352         -0.2094387        -0.22938827
## 932       -0.143050560         0.56471247          0.2390177        -0.35822483
## 3297      -0.084656215         0.27101245          0.4385655        -0.38487294
## 4685      -0.001167767         0.25425552          0.2302894        -0.05280242
##      0.3423823824497884 0.4884630853337104 0.5984292087376138
## 766          0.22332359          0.7026936          0.7322760
## 932         -0.01996294          0.5141617          0.4820379
## 3297         0.02122035          0.5030699          0.5516945
## 4685         0.25499532          0.6684735          0.6436055
##      0.2927636511301089 0.233825552475406 0.36030610067167634 0.304413462625733
## 766          0.16166959       0.114595065           0.6154359        0.57109047
## 932         -0.05554091      -0.665894943          -0.3643267       -0.46780941
## 3297         0.01922238      -0.483129527          -0.2538520       -0.20417184
## 4685         0.24238941      -0.008344068           0.1783239        0.08771943
##      0.511610836277837 0.5540874214525923 0.0965652318072868
## 766         1.53259836          1.4530432          0.1454652
## 932        -0.18501283         -0.2131731         -0.5575185
## 3297        0.07671401          0.1224073         -0.5014262
## 4685        0.39157866          0.3438343         -0.2467660
##      0.31740257140675465 0.2637698847445678 0.136338877840614
## 766           0.67631728          0.5731190        -0.2465296
## 932          -0.15509407         -0.2828990        -0.1495542
## 3297         -0.25874400         -0.2304479        -0.3717523
## 4685         -0.07882122         -0.1073084        -0.3062507
##      0.25932811026538605 0.2129808950361875 0.0914792140057591
## 766           0.14227247         0.12780586         -0.4383687
## 932           0.28305838         0.09462879         -0.2155773
## 3297         -0.02974996         0.02702020         -0.1357733
## 4685         -0.20404414        -0.24920646         -0.2064846
##      0.29587779632116323 0.33886632613890294 0.5180620591233701
## 766           -0.2574716         -0.03042826         -0.12623227
## 932            0.4637000          1.34023165          1.20275957
## 3297           0.2873934          0.90564030          1.06148140
## 4685          -0.1287809          0.06737074         -0.04009712
##      0.2184925118603042 0.21441932344455 0.4861313922407195 0.4457566445177248
## 766          -0.3377749       -0.5280206         -0.4390830         -0.4989106
## 932           0.2688001       -0.1366825          0.4688820          0.2188778
## 3297          0.2776403       -0.1696417          0.2139648          0.2296759
## 4685         -0.2091097       -0.2311721         -0.1165452         -0.1557444
##      0.8759406782194549 0.9236911811391476 0.398111926622587 0.5448712426120207
## 766         -0.41214975         -0.4542910        -0.4996249        -0.27377372
## 932          1.08063786          1.0221858        -0.2731893         0.16652692
## 3297         0.77366543          0.8071778        -0.4592769        -0.07340191
## 4685         0.03736671         -0.0423044        -0.2159622        -0.14333913
##      0.6209457850670713 0.155173850989368 0.45114115763541357
## 766         -0.32237981        -0.1087224          0.14899284
## 932          0.12454799        -0.4617624         -0.06233065
## 3297        -0.05891607        -0.4269254         -0.20620763
## 4685        -0.18612001        -0.2646038         -0.19376525
##      0.4161490353680438 0.150082533835292 0.3123944639615827 0.4501608697992175
## 766          0.08711755       -0.08223605          0.3635828         1.14559167
## 932         -0.21000537       -0.47628831         -0.1139123         0.12407645
## 3297        -0.20579989       -0.47699004         -0.2217123         0.09272154
## 4685        -0.21594051       -0.26964570         -0.2059457        -0.04500520
##      0.5623950879002614 0.2981183056877373
## 766           0.9324630          0.3238018
## 932           0.3120156         -0.2627014
## 3297          0.1411859         -0.2115363
## 4685         -0.2194457         -0.2390253
```

```
#extracting the time series of each gene

#for each selected gene, expression values are extracted
# i.e. all columns except the ID column
#then converting them to numeric to ensure the correct plotting
gene766  <- as.numeric(selected_data[selected_data$ID == 766,  -1])
gene932  <- as.numeric(selected_data[selected_data$ID == 932,  -1])
gene3297 <- as.numeric(selected_data[selected_data$ID == 3297, -1])
gene4685 <- as.numeric(selected_data[selected_data$ID == 4685, -1])



# plotting the Expression time series of the four genes
#starting by plotting the gene 766 alone
```

```r
# the y axis limits are defined using all four genes so that
# the curves fit together on the same scale.
plot(gene766, type = "l", col = "blue", lwd = 2,
     ylim = range(selected_data[,-1], na.rm = TRUE),
     xlab = "Timepoint", ylab = "Expression",
     main = "Expression of Genes 766, 932, 3297, 4685")

#then adding the remaining three gene curves using lines()
lines(gene932,  col = "red",       lwd = 2)
lines(gene3297, col = "darkgreen", lwd = 2)
lines(gene4685, col = "black",    lwd = 2)

par(xpd = NA) #allowing drawing outside plot
# A legend is added to indicate which color corresponds to which gene
legend("topright", inset = c(-0.095, 0),
       legend=c("766","932","3297","4685"),
       col=c("blue", "red", "darkgreen", "black"), lwd=2, pch=19,cex =0.8, bg = "white", bty = "o")
```



b. ***1.***

The code implements the peak-labelling procedure by assigning labels to time points according to their expression values and a chosen adjacency threshold. The function first determines the number of time points and creates a vector in which all positions are initially unlabeled. The expression values are then ordered from highest to lowest so that labeling begins at the largest local maxima. For each time point in this ordered list, neighbouring indices within the specified adjacency threshold are identified. If any neighbouring positions already carry a label, that label is inherited; otherwise, a new label is created. This procedure groups adjacent high-expression points into peak components.

2.

After all points are labelled, representative peak positions are extracted by selecting, within each label group, the time point with the maximum expression value. The function returns the complete set of labels as well as the final peak coordinates. The code is then applied to gene 766 with an adjacency threshold of 2, and the corresponding peak labels and representative peak positions are printed.

```r
# 2.2(b1) -

# This function groups timepoints into peaks based on:
# - their expression height
# - an adjacency threshold
# It returns a label for each timepoint and the peak centers.
label_peaks <- function(expression_values, adjacency_threshold = 1) {

  # Number of timepoints in the gene expression curve
  number_of_timepoints <- length(expression_values)

  # Starting with all timepoints unlabeled (-1)
  peak_labels <- rep(-1, number_of_timepoints)

  # Processing the timepoints from highest expression to lowest
  # (the idea is: assign labels starting from local maxima)
  order_from_highest <- order(expression_values, decreasing = TRUE)

  # Countering to assign new peak labels
  next_peak_label <- 0

  # Looping through timepoints in order of decreasing expression
  for (time_index in order_from_highest) {

    # Determining the neighboring timepoints within the threshold
    neighbor_indices <- (time_index - adjacency_threshold):(time_index + adjacency_threshold)
    neighbor_indices <- neighbor_indices[neighbor_indices >= 1 &
                                         neighbor_indices <= number_of_timepoints]

    # Checking if any neighbors already have a peak label
    neighbor_labels <- peak_labels[neighbor_indices]
    existing_labels <- neighbor_labels[neighbor_labels != -1]

    # If a neighbor has a label, inherit that label
    if (length(existing_labels) > 0) {
      peak_labels[time_index] <- existing_labels[1]

    # Otherwise, create a new peak label
    } else {
      peak_labels[time_index] <- next_peak_label
      next_peak_label <- next_peak_label + 1
    }
  }

# 2.2(b2) -
# A representative peak mrans the highest timepoint in each label group
  peak_positions <- sapply(unique(peak_labels), function(label_value) {
    indices_in_group <- which(peak_labels == label_value)
    indices_in_group[which.max(expression_values[indices_in_group])]
  })

  # Returning both: all labels AND the final peak positions
  return(list(
    peak_labels = peak_labels,
    peak_positions = peak_positions
  ))
}


test_gene <- gene766
peak_result <- label_peaks(test_gene, adjacency_threshold = 2)

cat("\nPeak Labels for Each Timepoint (Gene 766):\n")
```

```
##
## Peak Labels for Each Timepoint (Gene 766):
```

```
print(peak_result$peak_labels)
```

```
##  [1] 7 7 7 7 7 4 4 4 4 4 4 8 8 8 6 6 6 6 6 6 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 5
## [39] 5 5 5 5 5 9 9 9 3 3 3 3 3 3 3 1 1 1 1
```

```
cat("\nDetected Peak Positions (Gene 766):\n")
```

```
##
## Detected Peak Positions (Gene 766):
```

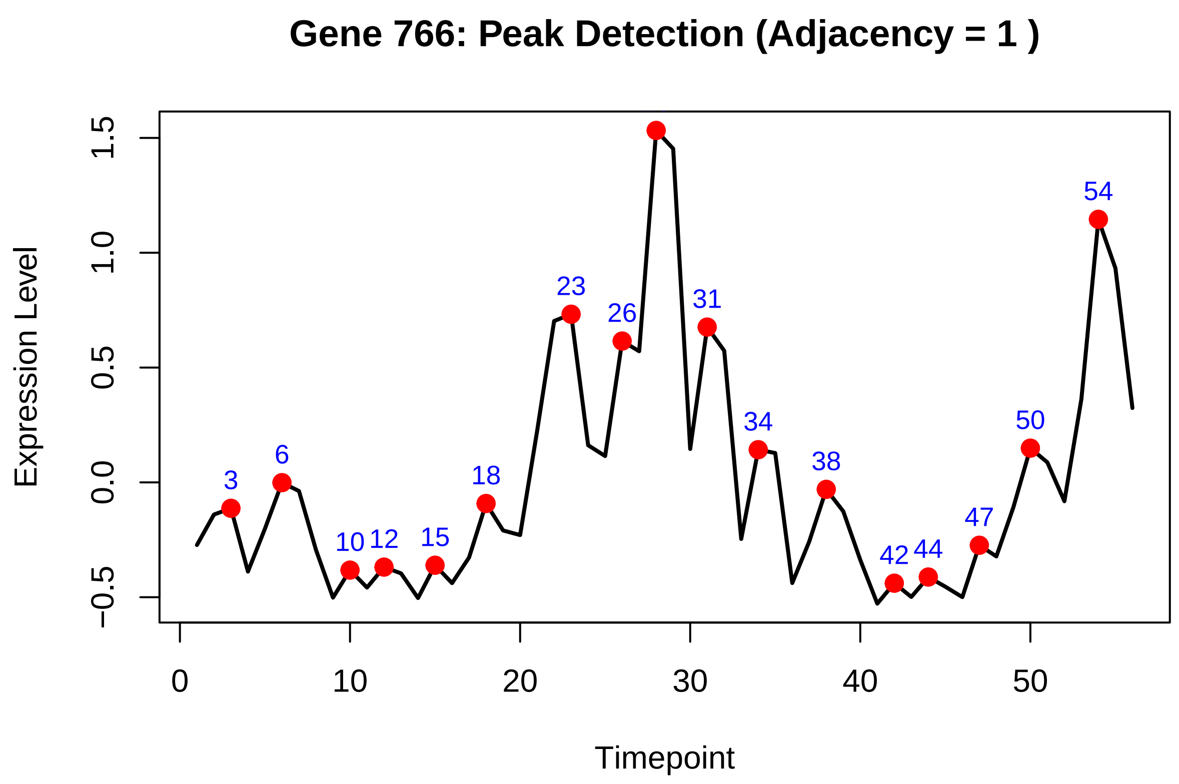
```
print(peak_result$peak_positions)
```

```
##  [1]  3  6 12 18 23 28 38 44 50 54
```

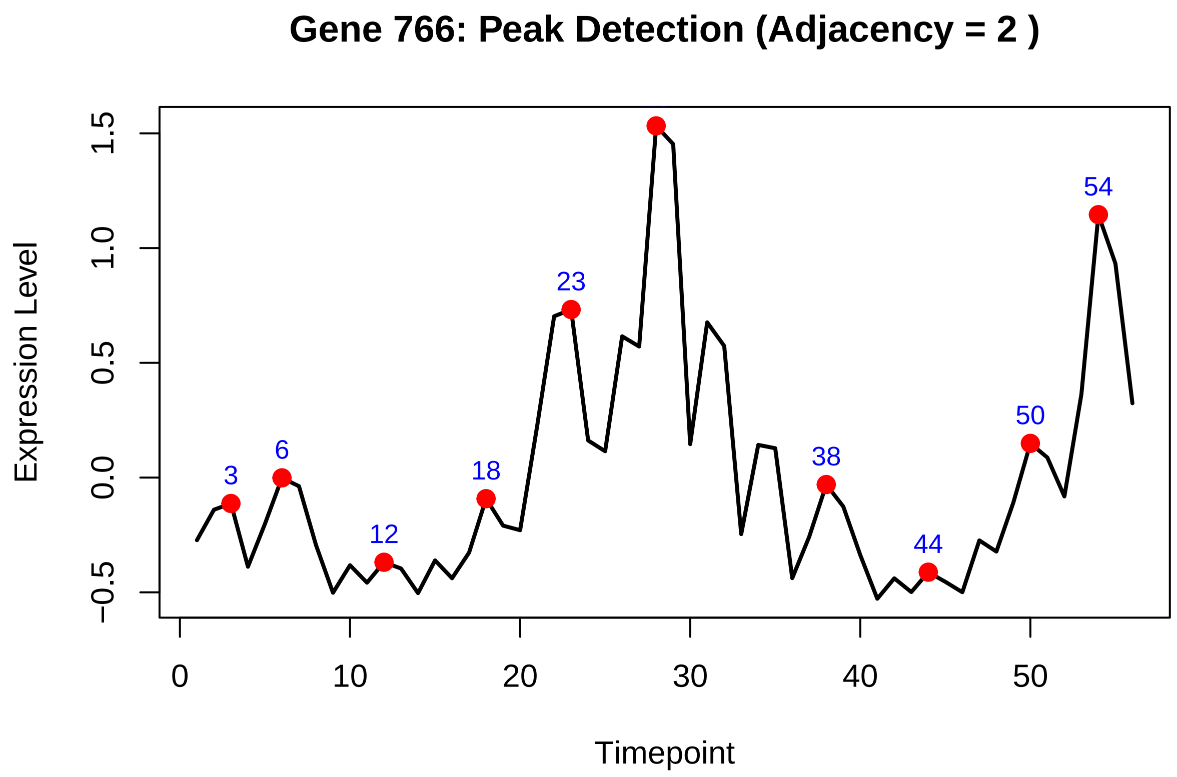
c.

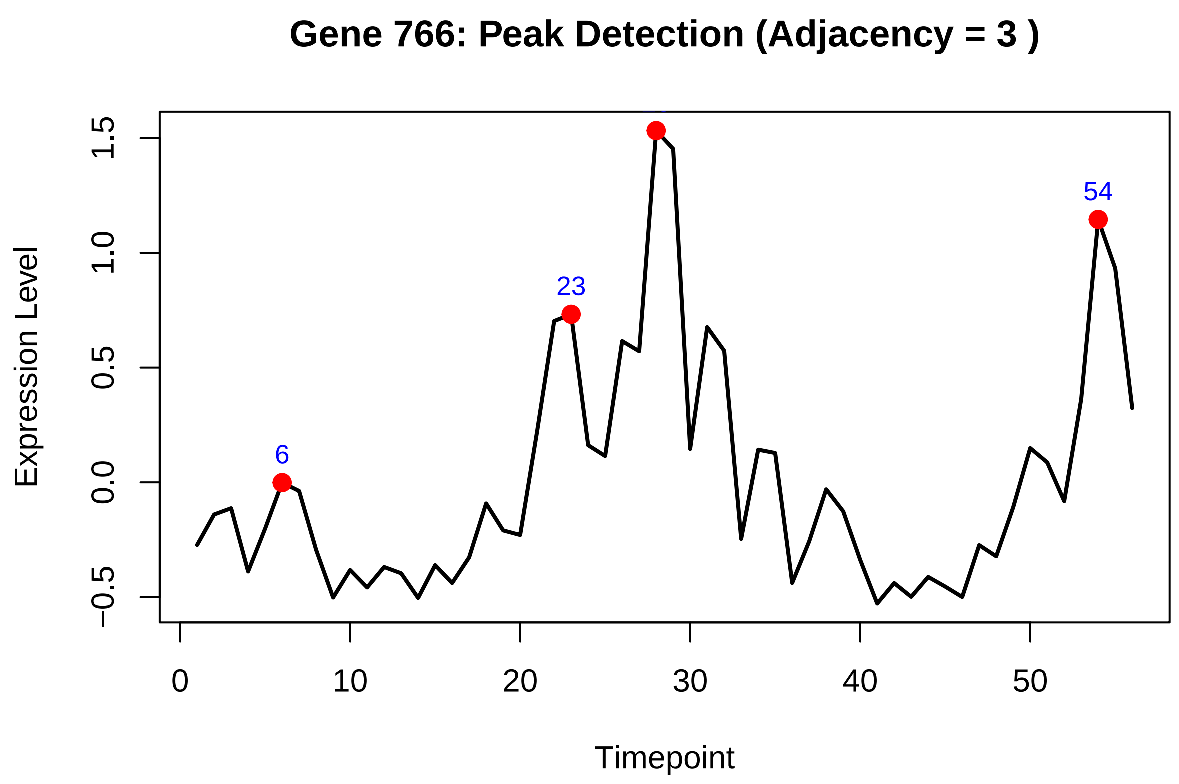
The code applies the previously defined peak-labelling function to gene 766 for thresholds 1–4. For each threshold, expression values are labelled, representative peak positions are extracted, and the peak coordinates are overlaid on the expression curve in a separate plot. This produces four visualisations showing how the number and position of detected peaks change as the adjacency threshold increases.

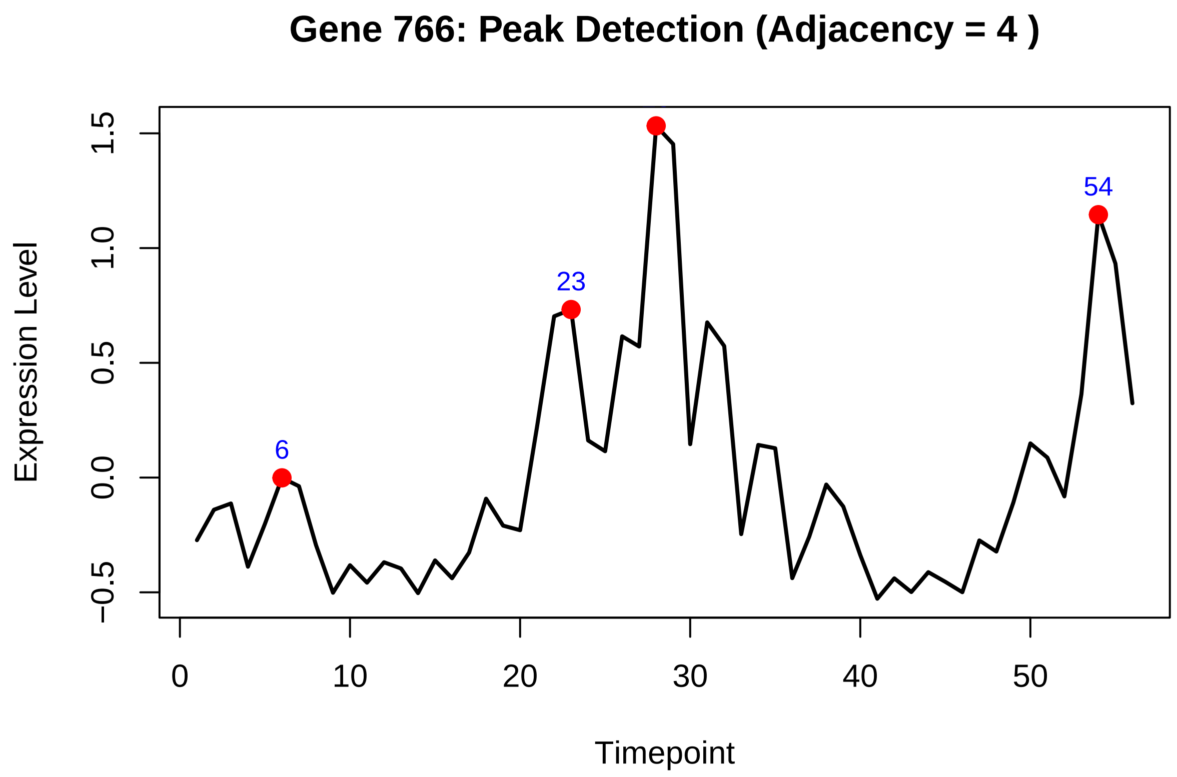
```
# Vector of thresholds to test
adjacency_values <- c(1, 2, 3, 4)

# Extracting expression values of gene 766
expression_766 <- gene766

# Loop over all adjacency thresholds
for (adj in adjacency_values) {
  # Run the peak detection function for this threshold
  peak_output <- label_peaks(expression_766, adjacency_threshold = adj)

  # Extracting the detected peak positions
  detected_peaks <- peak_output$peak_positions

  # Creating a plot for this threshold
  plot(expression_766, type = "l", lwd = 2, col = "black",
       xlab = "Timepoint",
       ylab = "Expression Level",
       main = paste("Gene 766: Peak Detection (Adjacency =", adj, ")"))

  # Adding peak markers
  points(detected_peaks,
         expression_766[detected_peaks],
         col = "red",
         pch = 19,
         cex = 1.2)

  # Labeling the peaks
  text(detected_peaks,
       expression_766[detected_peaks],
       labels = detected_peaks,
       pos = 3,
       cex = 0.8,
       col = "blue")
}
```



Gene 766: Peak Detection (Adjacency = 1)
Expression Level
Timepoint
−0.5
0.0
0.5
1.0
1.5
0
10
20
30
40
50
3
6
10
12
15
18
23
26
31
34
38
42
44
47
50
54


Gene 766: Peak Detection (Adjacency = 2)
Expression Level
Timepoint
−0.5
0.0
0.5
1.0
1.5
0
10
20
30
40
50
3
6
12
18
23
38
44
50
54


Gene 766: Peak Detection (Adjacency = 3)
Expression Level
Timepoint
6
23
54
−0.5
0.0
0.5
1.0
1.5
0
10
20
30
40
50



The detected peak structures varied substantially across adjacency thresholds. With an adjacency threshold of 1, the algorithm produced many narrow peaks because only directly neighbouring time points were grouped together. This caused small fluctuations in expression to be interpreted as separate peak events.

Increasing the threshold to 2 reduced the number of detected peaks and yielded clearer peak components, as closely spaced fluctuations were merged into coherent structures. When the threshold was increased further to 3 and 4, the results became nearly identical, with only a small set of broad peak regions remaining. At these higher thresholds, the algorithm merged most local variations, retaining only the major high-expression regions.

Overall, an adjacency threshold of 2 appears best suited for the expression pattern of gene 766. This parameter balances sensitivity and peak stability by suppressing noise-driven fluctuations while still distinguishing biologically meaningful peaks that become overly merged at larger thresholds.

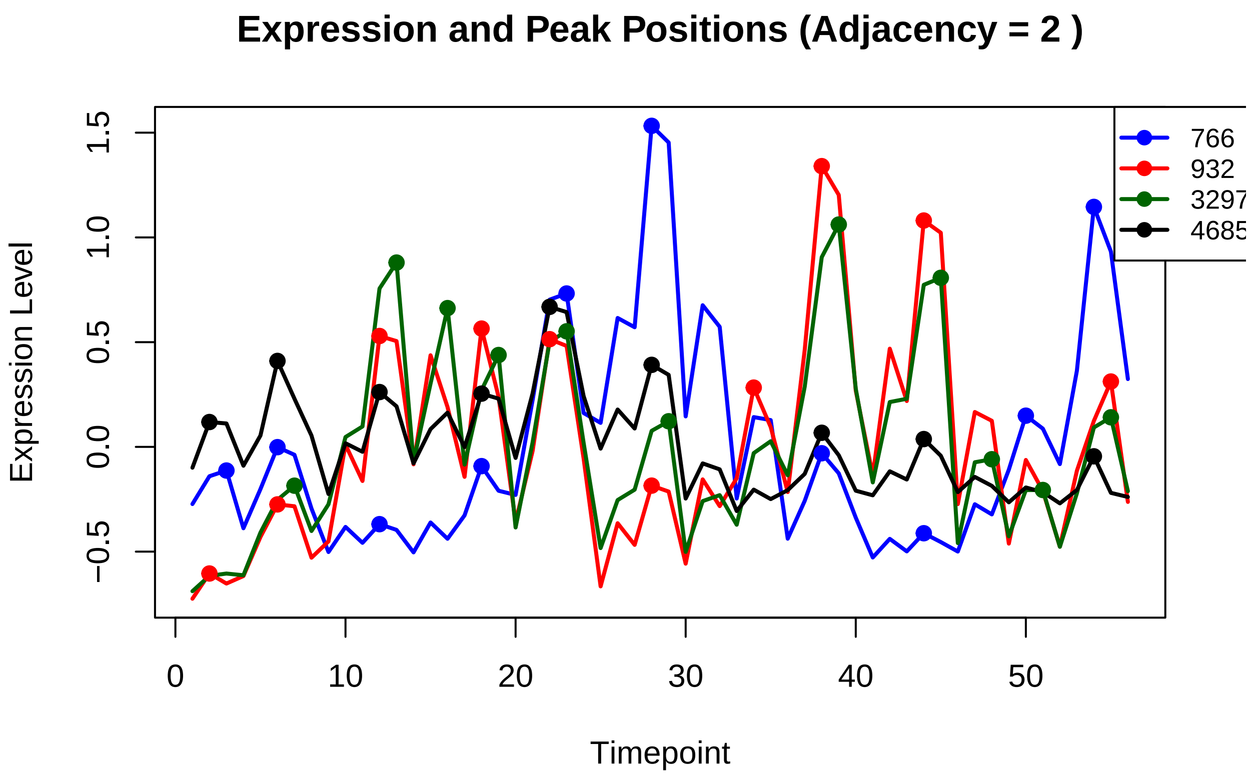
d.

Peak detection was performed for each of the four genes using an adjacency threshold of two time points, a value chosen based on the behaviour observed in part (c). The peak-labeling function was applied separately to each gene, producing the group labels and representative peak coordinates.

A combined plot was generated by first drawing the expression profile of one gene to initialise the plotting window, followed by overlaying the expression curves of the remaining genes. The detected peak positions for each gene were then added as coloured point markers, with consistent colour assignments between the curves and the peak highlights. A legend was included to support interpretation of the four overlaid expression trajectories.

```r
# Peak detection for all four genes
# Using a reasonable adjacency threshold (chosen: 2)
adjacency_setting <- 2
```

```r
# Detecting peaks for each gene
peaks_766  <- label_peaks(gene766,  adjacency_threshold = adjacency_setting)
peaks_932  <- label_peaks(gene932,  adjacency_threshold = adjacency_setting)
peaks_3297 <- label_peaks(gene3297, adjacency_threshold = adjacency_setting)
peaks_4685 <- label_peaks(gene4685, adjacency_threshold = adjacency_setting)


# combined plot all genes with their peaks

plot(gene766, type="l", col="blue", lwd=2,
     ylim = range(selected_data[,-1], na.rm = TRUE),
     xlab="Timepoint", ylab="Expression Level",
     main=paste("Expression and Peak Positions (Adjacency =", adjacency_setting, ")"))

lines(gene932,  col="red",       lwd=2)
lines(gene3297, col="darkgreen", lwd=2)
lines(gene4685, col="black",  lwd=2)

# Adding the peak markers for each gene
points(peaks_766$peak_positions,  gene766[peaks_766$peak_positions],
       col="blue", pch=19)
points(peaks_932$peak_positions,  gene932[peaks_932$peak_positions],
       col="red", pch=19)
points(peaks_3297$peak_positions, gene3297[peaks_3297$peak_positions],
       col="darkgreen", pch=19)
points(peaks_4685$peak_positions, gene4685[peaks_4685$peak_positions],
       col="black", pch=19)

par(xpd = NA) #allowing drawing outside plot
# Adding legend
legend("topright", inset = c(-0.095, 0),
       legend=c("766","932","3297","4685"),
       col=c("blue", "red", "darkgreen", "black"), lwd=2, pch=19,cex =0.8, bg = "white", bty = "o")
```



Yes. The expression profiles of the four genes show a consistent pattern of periodic behaviour. Across the measured time points, each gene exhibits repeated phases of up-regulation followed by down-regulation, indicating that their transcription is not constant but instead follows a cyclic pattern. Such recurring regulatory changes are characteristic of genes whose activity is coordinated with the cell cycle. The periodic structure of the expression curves therefore suggests that these genes are controlled in a phase-dependent manner, becoming up-regulated when their function is required and down-regulated once that stage of the cycle has passed.

The progression of the detected peaks shows that the four genes do not reach maximal expression at the same time but instead display shifted activation across the cell-cycle time course. This temporal separation indicates that the genes operate in different regulatory phases. Gene 766 peaks earliest, suggesting involvement in early cell-cycle processes, while genes 932 and 3297 peak later, consistent with mid-cycle activity. Gene 4685 shows its highest expression toward the end of the cycle, implying a late-phase role. Overall, the sequential arrangement of peaks reflects phase-specific gene activation characteristic of cell-cycle regulation.